Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method for processing a complex request and to optimize the number of the SNMP requests transmitted through a network, wherein the complex request is addressed to at least one SNMP agent of a resource machine of a computer system from a complex protocol manager of an application machine, the application and resource machines communicating through a network, each agent managing one or more attribute tables belonging to the resource machine, the instances of the tables being referenced by identifiers comprising indexes, the method ~~comprising~~consisting of:

- transforming a first filter derived from a complex request from the manager of the application machine into a second simplified filter comprising only conditions on indexes, the second simplified filter corresponding to the following matching characteristics: the second simplified filter lets through all the SNMP requests whose responses could verify the first filter, based on conditions whose attribute values could verify the first filter, and the second simplified filter filters out all the SNMP requests whose responses cannot in any way verify the first filter because the conditions on indexes associated with said filtered-out SNMP requests do not verify the first filter regardless of attribute values associated with said conditions;

wherein the transforming further comprises deleting from the first filter all conditions that operate on attributes that are not associated with any of said indexes;

- limiting the SNMP requests to those that comply with the second simplified filter;

- transmitting said limited SNMP requests to the SNMP agent of the resource machine through the network; and

applying the first filter to the responses obtained to the SNMP requests,

whereby the method makes it possible to process said complex request and to optimize the number of the SNMP requests transmitted through the network.

2. (Previously Presented) The method according to claim 1, further comprising:

determining a first potential instance that verifies the second simplified filter , based on a test identifier that is less than an identifier of the first potential instance;

finding, using an SNMP request, a solution instance of the table having as its identifier an identifier that is subsequent to the test identifier;

terminating further processing if no solution instance is found;

applying the first filter, said first filter comprising a complex filter, to the solution instance;

determining whether or not the solution instance verifies the first filter;

determining whether or not the first potential instance whose identifier is higher than the identifier of the solution instance and that verifies the second simplified filter ;

terminating further processing if the solution instance does not verify the second simplified filter; and

if the solution instance verifies the second simplified filter, setting as the test identifier an identifier that is less than the identifier of the first potential.

3. (Canceled)

4. (Previously Presented) The method according to claim 2, further comprising:

scanning the table in its entirety if the second simplified filter includes only the TRUE condition; and

wherein no instance can verify the second simplified filter if the second simplified filter includes only the FALSE condition.

5. (Previously Presented) The method according to claim 1, said transforming the first filter into the second simplified filter comprises immediately verifying whether the first filter responds to rules defining filters that are not verified by any instance, said first filter being a complex filter.

6. (Currently Amended) The method according to claim 1, wherein said transforming the first filter into the second simplified filter comprises:

- transforming the first filter into a combination of conditions on the attributes joined by the logical operators [[HAND]]AND, OR and NOT, the first filter being a complex filter;
- pushing the NOT operators to to outer portions of a tree representation of the second simplified filter, and deleting occurrences of double NOTs (NOT NOT);
- deleting conditions affecting the attributes that are not indexes;
- simplifying the resulting operations by replacing operands;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying again.

7. (Previously Presented) The method according to claim 6, in which said deleting the conditions affecting the attributes that are not indexes comprisies replacing the conditions X and NOT X with the constant TRUE.

8. (Previously Presented) The method according to claim 6, in which said simplifying the resulting operationscomprises:

- replacing AND and OR tests associated with only one operand with the one operand;
- replacing AND operations containing only TRUE operands with the constant TRUE, and replacing OR operations containing only FALSE operands with the constant FALSE;
- removing TRUE conditions from other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with the constant TRUE, and replacing AND operations containing at least one FALSE operand with the constant FALSE;
- replacing conditions that are always TRUE or FALSE with the constant TRUE or FALSE; and
- repeating said simplifying the resulting operations until no further simplifying is possible.

9. (Previously Presented) The method according to claim 2, in which said determining a first potential instance that verifies the second filter comprises:

concatenating a value that verifies $C1_{(i)}$ with a first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain one or more zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}$. ... $In\_0_{(i)}$; and

selecting as the first potential instance the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

10. (Previously Presented) The method according to claim 9, further comprising:

performing, for any i and as long as an index p is greater than 0, or as long as no instance searched for has been found, the following operations

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed by

- for any index $k < p$, take the value Ik with I1.I2. ... .In being the identifier of the solution instance;
- for the index p, take the value $Jp_{(i)}$; and
- for any index $k > p$, take the value $Ik\_0_{(i)}$;

Otherwise, p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

11. (Previously Presented) The method according to claim 2, in which obtaining the test identifier from the identifier of the potential instance is performed by subtracting one from a last number of the test identifier if the identifier of the potential instance is different from 0, or by deleting the last number if the last number is null.

12. (Currently Amended) A system for processing complex requests and for optimizing the number of the SNMP requests transmitted through a network, the complex request addressed to at least one SNMP agent of a resource machine of a computer system from a complex protocol manager of an application machine, each said agent being configured to manage one or more attribute tables associated with the resource machine, in which instances of the tables are referenced by identifiers comprising indexes, the system ~~comprising~~consisting of:

an integrating agent configured to transform a first filter derived from a complex request from the manager of the application machine into a second simplified filter comprising only conditions on indexes, the second simplified filter corresponding to the following matching characteristics: the second simplified filter lets through all the SNMP requests whose responses could verify the first filter, based on conditions whose attribute values could verify the first filter, and the second simplified filter filters out all the SNMP requests whose responses cannot in any way verify the first filter because the conditions on indexes associated with said filtered-out SNMP requests do not verify the first filter regardless of attribute values associated with said conditions;

wherein the transforming further comprises deleting from the first filter all conditions that operate on attributes that are not associated with any of said indexes; and

the integrating agent further configured to limit the SNMP requests to those that comply with the second simplified filter, to transmit said limited SNMP requests to the SNMP agent of the resource machine through the network, and to apply the first filter to the responses obtained to the SNMP requests,

whereby the system makes it possible to process said complex request and to optimize the number of the SNMP requests transmitted through the network.

13. (Currently Amended) A method for processing a complex request and to optimize the number of the SNMP requests transmitted through a network, wherein the complex request is addressed to at least one SNMP agent [[(5)]]of a resource machine ~~(2b)~~ of a computer system [[(1)]]from a complex protocol manager [[(4)]]of an application machine ~~(2a)~~, wherein the complex request addressed to the agent [[(5)]]from the manager [[(4)]]comprises SNMP attributes managed by the agent [[(5)]]and capable of being represented by a filter ~~(F1, F2)~~ constituted by any number of conditions on any number of attributes, linked to one another by any number of Boolean operators (AND, OR, NOT, and EX.OR~~, etc.~~) and the application ~~(2a)~~ and resource ~~(2b)~~ machines communicate through a network[[(3)]], each agent [[(5)]]managing attribute tables belonging to the resource machine ~~(2b)~~, the instances of the tables being referenced by identifiers comprising indexes, ~~comprising~~consisting of:

- transforming a complex filter ~~(F1)~~ derived from the complex request addressed to agent [[(5)]]from the manager [[(4)]]of the application machine ~~(2a)~~ into a simplified filter ~~(F2)~~ comprising only conditions on indexes, the simplified filter ~~(F2)~~ adapted to let through all the SNMP requests whose responses could verify the complex filter ~~(F1)~~, based on conditions whose attribute values could verify the complex filter ~~(F1)~~, and to filter out all the SNMP requests whose responses cannot in any way verify the complex filter ~~(F1)~~ because the conditions on indexes associated with said filtered-out SNMP requests do not verify the first filter ~~(F1)~~ regardless of attribute values associated with said conditions;

wherein the transforming further comprises deleting from the complex filter ~~(F1)~~ all conditions that operate on attributes that are not associated with any of said indexes;

- limiting the SNMP requests to those that comply with the simplified filter ~~(F2)~~;

- transmitting said limited SNMP requests to the SNMP agent [[(5)]]of the resource machine ~~(2b)~~ through the network[[(3)]]; and

applying the complex filter ~~(F1)~~ to the responses obtained to the SNMP requests, whereby the method makes it possible to process said complex request and to optimize the number of the SNMP requests transmitted through the network.

14. (Currently Amended) A method according to claim 13, wherein an identifier just below an identifier of the potential instance determined is a test identifier, the method further comprising:

1) determining a first potential instance that verifies the simplified filter ~~(F2)~~;

2) using an SNMP request to find an instance of the table having as its identifier an identifier that follows a test identifier and if no instance of the table is found, terminating processing of the method, and if an instance is found, naming the instance found a solution instance;

3) determining whether the solution instance is part of the response to the complex request processed by verifying the complex filter ~~(F1)~~ and upon verification of the complex filter ~~(F1)~~, applying the complex filter ~~(F1)~~ to the solution instance; and

4) determining the first potential instance whose identifier is higher than the identifier of the solution instance and that verifies the simplified filter ~~(F2)~~ and terminating processing of the method if no instance is found, and if an instance is found, naming the identifier that is just below the identifier of the potential instance a test identifier and resuming the step of using the SNMP request to find the instance of the table having as its identifier the identifier that follows the test identifier.

15. (Canceled)

16. (Currently Amended) A method according to claim 14, wherein in the first step, after simplification, the simplified filter ~~(F2)~~ is reduced to:

- only the TRUE condition, in which case the table is scanned in its entirety; and
- only the FALSE condition, in which case no instance can work.

17. (Currently Amended) A method according to claim 15, wherein in the first step, after simplification, the simplified filter ~~(F2)~~ is reduced to:

- only the TRUE condition, in which case the table is scanned in its entirety; and
- only the FALSE condition, in which case no instance can work.

18. (Currently Amended) The method according to claim 14, further comprising: in order to obtain the simplified filter[[F2]], immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

19. (Currently Amended) The method according to claim 15, further comprising: in order to obtain the simplified filter[[F2]], immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

20. (Currently Amended) The method according to claim 16, further comprising: in order to obtain the simplified filter[[F2]], immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

21. (Currently Amended) The method according to claim 17, further comprising: in order to obtain the simplified filter[[F2]], immediately verifying whether the complex filter responds to rules defining filters that are not verified by any instance.

22. (Currently Amended) The method according to claim 13, further comprising:

in order to obtain[[a]]the simplified filter[[F2]],

- transforming the complex filter ~~(F1)~~ into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves of a tree representing the simplified filter and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

23. (Currently Amended) The method according to claim 14, further comprising:

in order to obtain[[a]]the simplified filter[[F2]],

- transforming the complex filter (F1) into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;

- pushing NOT operators to the leaves of a tree representation of the simplified filter and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

24. (Currently Amended) A method according to claim 15, further comprising:

in order to obtain[[a]]the simplified filter[[F2]],

- transforming the complex filter ~~(F1)~~ into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves of a tree representation of the simplified filter and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

25. (Currently Amended) The method according to claim 16, further comprising:

in order to obtain[[a]]the simplified filter[[F2]],

- transforming the complex filter ~~(F1)~~ into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves of a tree representation of the simplified filter and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

26. (Currently Amended) A method according to claim 17, further comprising:

in order to obtain[[a]]the simplified filter[[F2]],

- transforming the complex filter ~~(F1)~~ into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves of a tree representation of the simplified filter and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

27. (Currently Amended) A method according to claim 18, further comprising:

in order to obtain[[a]]the simplified filter[[F2]],

- transforming the complex filter ~~(F1)~~ into a combination of conditions on the attributes joined by the logical operators AND, OR and NOT;
- pushing NOT operators to the leaves of a tree representation of the simplified filter and deleting double NOTs (NOT NOT);
- deleting the conditions X affecting the attributes that are not indexes;
- simplifying the resulting operations;
- factoring the nested ANDs and ORs;
- gathering the conditions related to the same index; and
- gathering all the ORs at the route of the filter and simplifying the resulting operations again.

28. (Previously Presented) A method according to claim 22, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

29. (Previously Presented) A method according to claim 23, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

30. (Previously Presented) A method according to claim 24, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

31. (Previously Presented) A method according to claim 25, comprising replacing the conditions X and NOT X with the constant TRUE in order to delete the conditions X.

32. (Previously Presented) The method according to claim 18, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

33. (Previously Presented) The method according to claim 23, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;

- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

34. (Previously Presented) The method according to claim 24, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;

- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

35. (Previously Presented) The method according to claim 25, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

36. (Previously Presented) The method according to claim 26, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

37. (Previously Presented) The method according to claim 27, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;

- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

38. (Previously Presented) The method according to claim 28, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE ,and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

39. (Previously Presented) The method according to claim 29, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

40. (Previously Presented) The method according to claim 30, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;

- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

41. (Previously Presented) The method according to claim 31, having AND and OR operations and comprising further simplifying operations, the method comprising:

- replacing AND and OR operations having only one operand with said one operand;
- replacing AND operations containing only TRUE operands with a constant TRUE, and replacing OR operations containing only FALSE operands with a constant FALSE;
- removing TRUE conditions from the other AND operations, and removing FALSE conditions from the other OR operations;
- replacing OR operations containing at least one TRUE operation with a constant TRUE, and replacing AND operations containing at least one FALSE operand with a constant FALSE;
- replacing conditions that are always TRUE with a constant TRUE, and replacing conditions that are always FALSE with a constant FALSE; and

repeating each said further simplifying operation as many times as it is possible to do so.

42. (Previously Presented) The method according to claim 14, wherein the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

43. (Previously Presented) The method according to claim 15, wherein the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

44. (Previously Presented) The method according to claim 16, wherein the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

45. (Previously Presented) The method according to claim 18, wherein the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

46. (Previously Presented) The method according to claim 22, wherein the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

47. (Previously Presented) The method according to claim 28, wherein the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$, the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

48. (Previously Presented) The method according to claim 32, wherein the step of determining the first potential instance that verifies the simplified filter comprises concatenating the first value that verifies $C1_{(i)}$ with the first value that verifies $C2_{(i)}$, and so on up to $Cn_{(i)}$, in order to obtain zero local potential instances $I1\_0_{(i)}.I2\_0_{(i)}. \ldots In\_0_{(i)}$,

the first possible value without a condition on a given index being the null value, the potential instance corresponding to the smallest of the zero local potential instances.

49. (Previously Presented) The method according to claim 42, wherein the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index $k < p$, we take the value Ik with I1.I2. ... .In being the identifier of the solution instance;
- for the index p, we take the value $Jp_{(i)}$; and
- for any index $k > p$, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

50. (Previously Presented) The method according to claim 43, wherein the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index $k < p$, we take the value Ik with I1.I2. ... .In being the identifier of the solution instance;

- for the index p, we take the value $Jp_{(i)}$; and
- for any index $k > p$, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

51. (Previously Presented) The method according to claim 44, wherein the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index $k < p$, we take the value Ik with I1.I2. ... .In being the identifier of the solution instance;
- for the index p, we take the value $Jp_{(i)}$; and
- for any index $k > p$, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

52. (Previously Presented) The method according to claim 45, wherein the step of determining the first potential instance whose identifier is higher than the identifier of the solution instance comprises performing, for any i and as long as the index p is greater than 0 or as long as no instance searched for has been found, the following operations:

if there exists a $Jp_{(i)} > Ip$ that verifies the condition $Cp_{(i)}$, then the local potential instance is formed in the following way:

- for any index $k < p$, we take the value Ik with I1.I2. ... .In being the identifier of the solution instance;
- for the index p, we take the value $Jp_{(i)}$; and
- for any index $k > p$, we take the value $Ik\_0_{(i)}$;

otherwise p takes the value p-1 and the method repeats the above operations, the potential instance corresponding to the smallest of the local potential instances obtained.

53. (Previously Presented) The method according to claim 14, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

54. (Previously Presented) The method according to claim 15, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

55. (Previously Presented) The method according to claim 16, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

56. (Previously Presented) The method according to claim 18, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

57. (Previously Presented) The method according to claim 22, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

58. (Previously Presented) The method according to claim 28, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by

subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

59. (Previously Presented) The method according to claim 32, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

60. (Previously Presented) The method according to claim 42, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

61. (Previously Presented) The method according to claim 49, wherein the steps of determining the first potential instance that verifies the simplified filter and the first potential instance whose identifier is higher than the identifier of the solution instance consist of obtaining the test identifier from the identifier of the potential instance, by subtracting one from its last number if the latter is different from 0, or by deleting this last number if it is null.

62. (Currently Amended) A system for processing a complex request and to optimize the number of the SNMP requests transmitted through a network, comprising at least one SNMP agent[[(5)]] of a resource machine ~~(2b)~~ of a computer system[[(1)]] to which the complex request is transmitted from a complex protocol manager[[(4)]] of an application machine ~~(2a)~~, each agent[[(5)]] managing attribute tables belonging to the resource machine ~~(2b)~~, instances of the tables being referenced by identifiers comprising indexes, the system comprising an integrating agent[[(6)]] for processing the complex request,

means for transforming a complex filter ~~(F1)~~ derived from the complex request addressed to agent[[(5)]] from the manager[[(4)]] of the application machine ~~(2a)~~ into a simplified filter ~~(F2)~~ comprising only conditions on indexes, the simplified filter ~~(F2)~~ adapted to let through all SNMP requests whose responses could verify the complex filter ~~(F1)~~, based on conditions whose attribute values could verify the complex filter ~~(F1)~~, and to filter out all SNMP requests whose responses cannot in any way verify the complex filter ~~(F1)~~ because the conditions on indexes associated with said filtered-out requests do not verify the complex filter ~~(F1)~~ regardless of attribute values associated with said conditions;

wherein the transforming further comprises deleting from the complex filter ~~(F1)~~ all conditions that operate on attributes that are not associated with any of said indexes;

means for limiting SNMP requests to those that comply with the complex filter ~~(F2)~~;

means for transmitting said limited SNMP requests to the SNMP agent[[(5)]] of the resource machine ~~(2b)~~ through the network[[(3)]]; and

means for applying the simplified filter ~~(F1)~~ to the responses obtained to the SNMP requests.

whereby the method makes it possible to process said complex request and to optimize the number of the SNMP requests transmitted through the network.

63. (Currently Amended) The system for processing as set forth in claim 62 further comprising means for determining the first potential instance that verifies the simplified filter ~~(F2)~~ wherein the identifier first below the identifier of the potential instance determined is a test identifier.

64. (Currently Amended) The system for processing as set forth in claim 63 wherein, using an SNMP request, there is provided means to find the instance of the table having as its identifier the one that follows the test identifier and if no instance is found, terminating the processing method, if an instance is found, naming the instance found a solution instance; and means for determining whether the solution instance is part of the response to the complex request processed by verifying the complex filter ~~(F1)~~ and upon verification of the complex filter ~~(F1)~~, applying the complex filter ~~(F1)~~ to the solution instance.

65. (Canceled)